Istio new workload identity provision based envoy SDS

Quanjie Lin, Diem Vu

What is a service mesh?

A service mesh provides a transparent and language-independent way to flexibly and easily automate application network functions.

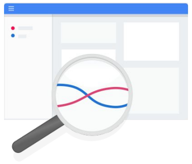
What is Istio?

A service mesh. But more: an open services platform to manage service interactions across container- and VM-based workloads

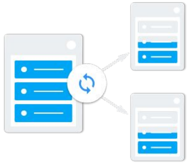
Istio Value Proposition

Securing service traffic

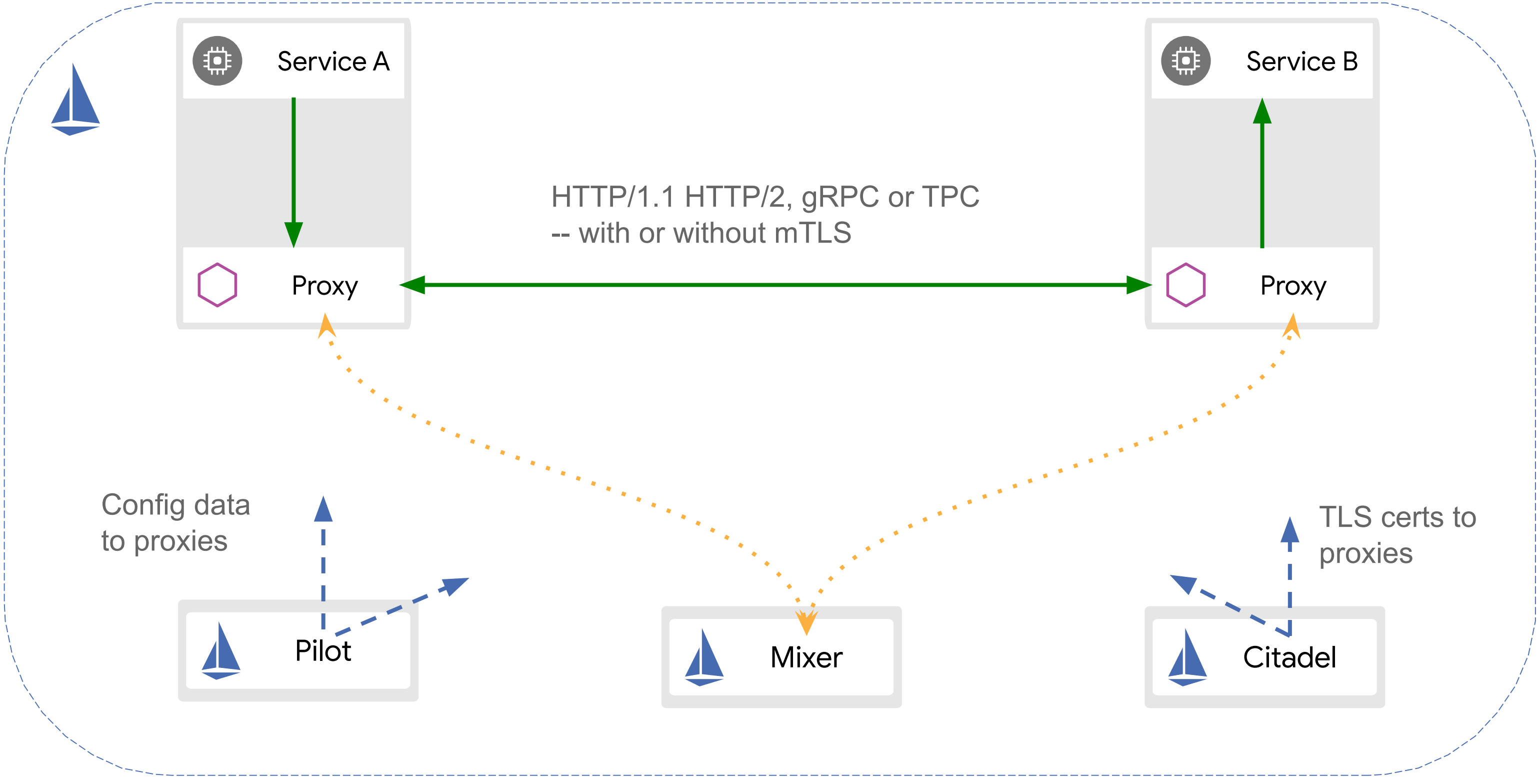
Uniform observability



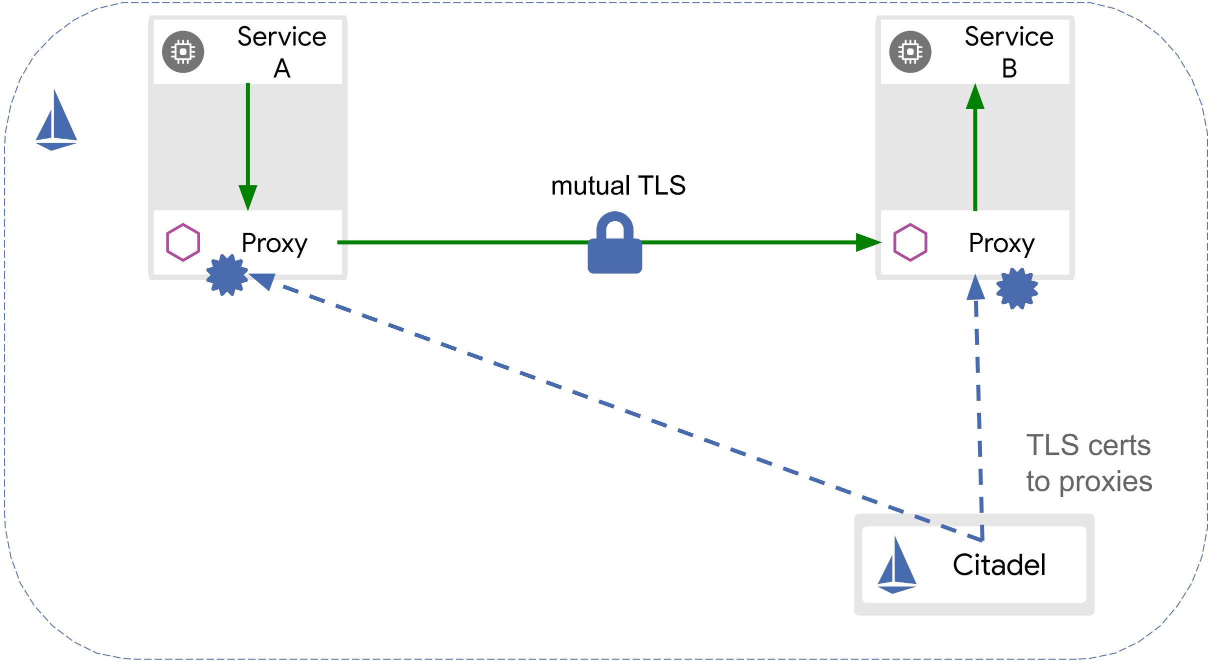
Operational agility



Istio overview



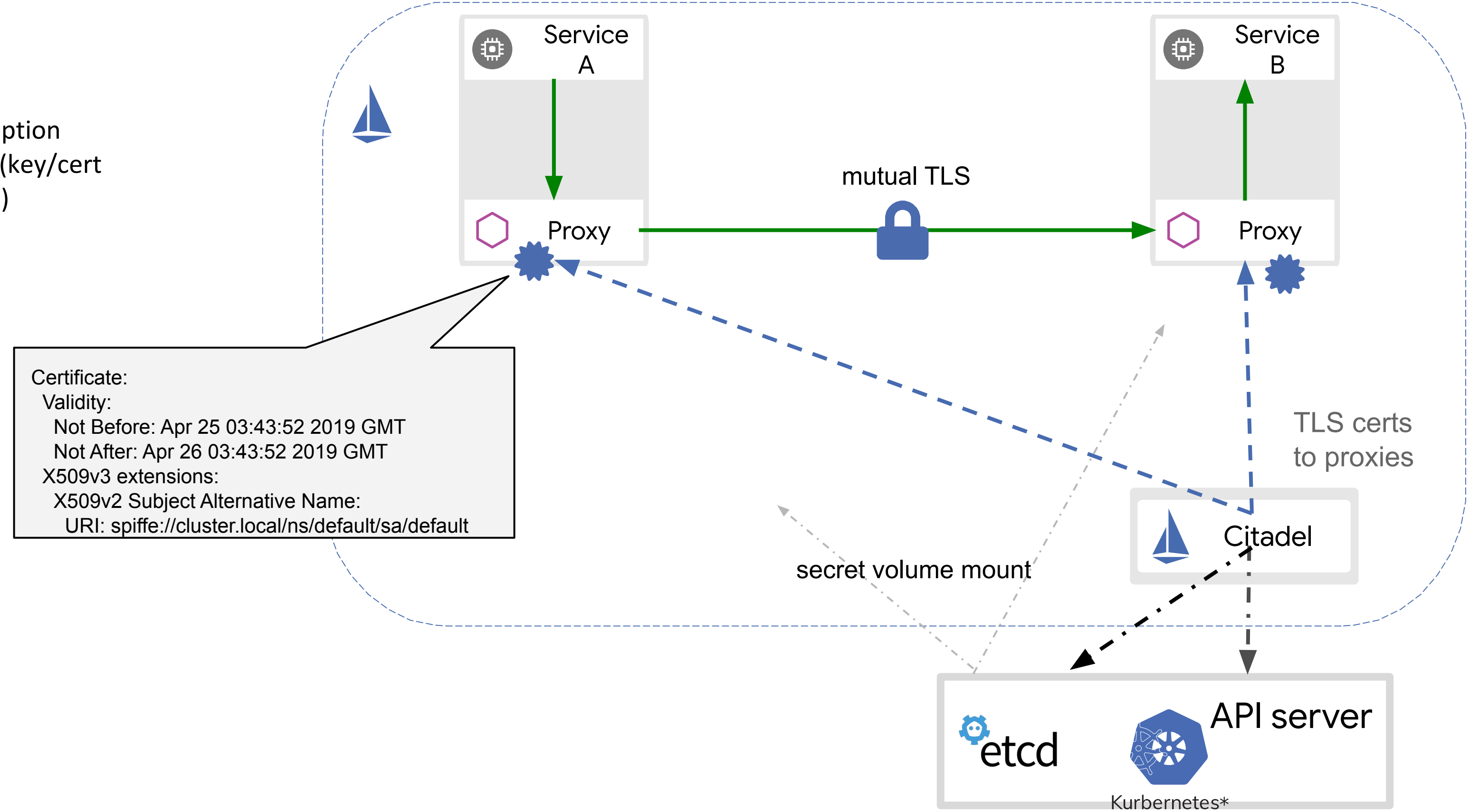
Identity provision pipeline (before Istio 1.1)



Identity provision pipeline (before Istio 1.1)

Problems

- ❏ Traffic interruption
- ❏ Security issue(key/cert stored as files)



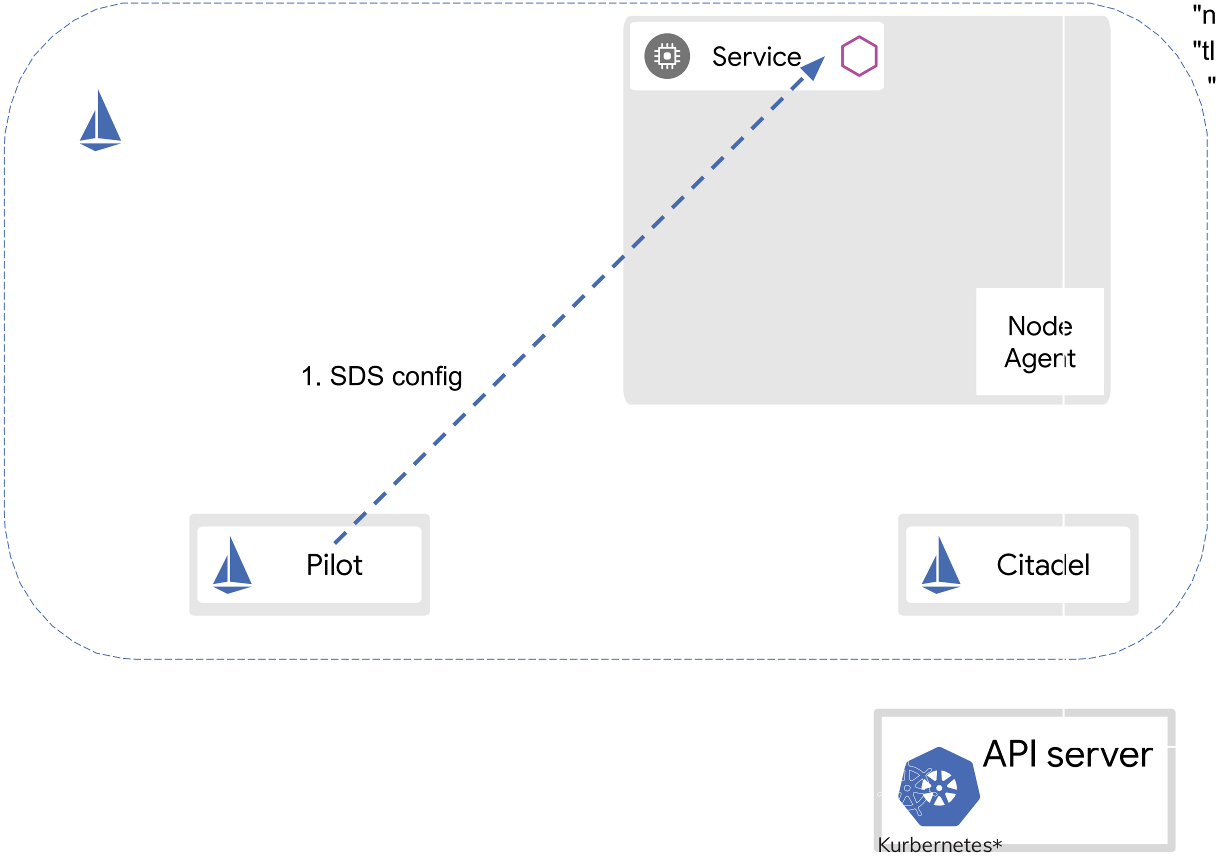
```
Certificate:
  Validity:
    Not Before: Apr 25 03:43:52 2019 GMT
    Not After: Apr 26 03:43:52 2019 GMT
  X509v3 extensions:
    X509v2 Subject Alternative Name:
      URI: spiffe://cluster.local/ns/default/sa/default
```

Envoy Secret Discovery Service(SDS)

- Enable envoy to fetch secret(key/certificate) dynamically from SDS server
- Simplify the certificate management
  - No traffic interruption during cert rotation
  - More secure(memory VS file system)
  - Plugable CA
- API interface

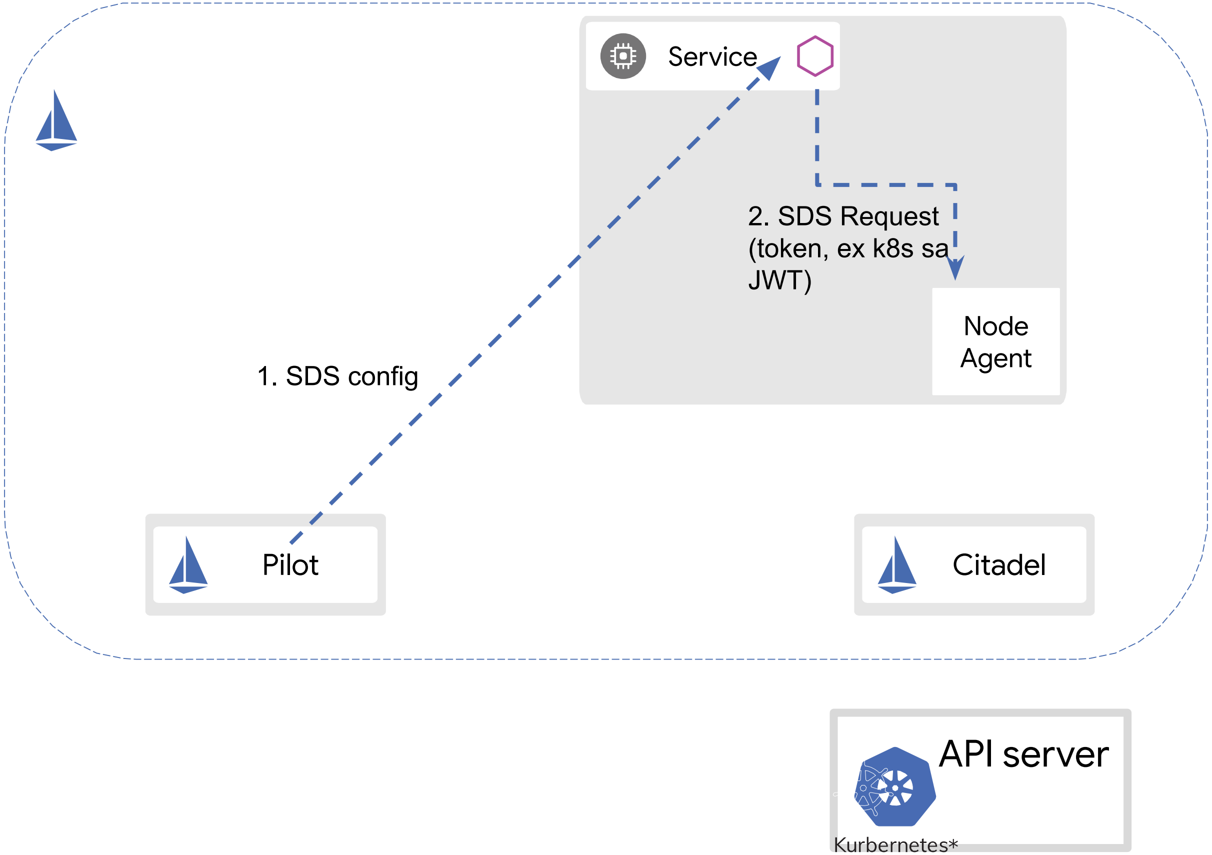
```
service SecretDiscoveryService {
  rpc StreamSecrets(stream envoy.api.v2.DiscoveryRequest)
      returns (stream envoy.api.v2.DiscoveryResponse) {
  }

  rpc FetchSecrets(envoy.api.v2.DiscoveryRequest) returns (envoy.api.v2.DiscoveryResponse) {
    option (google.api.http) = {
      post: "/v2/discovery:secrets"
      body: "*"
    };
  }
}
```

New identity provision pipeline

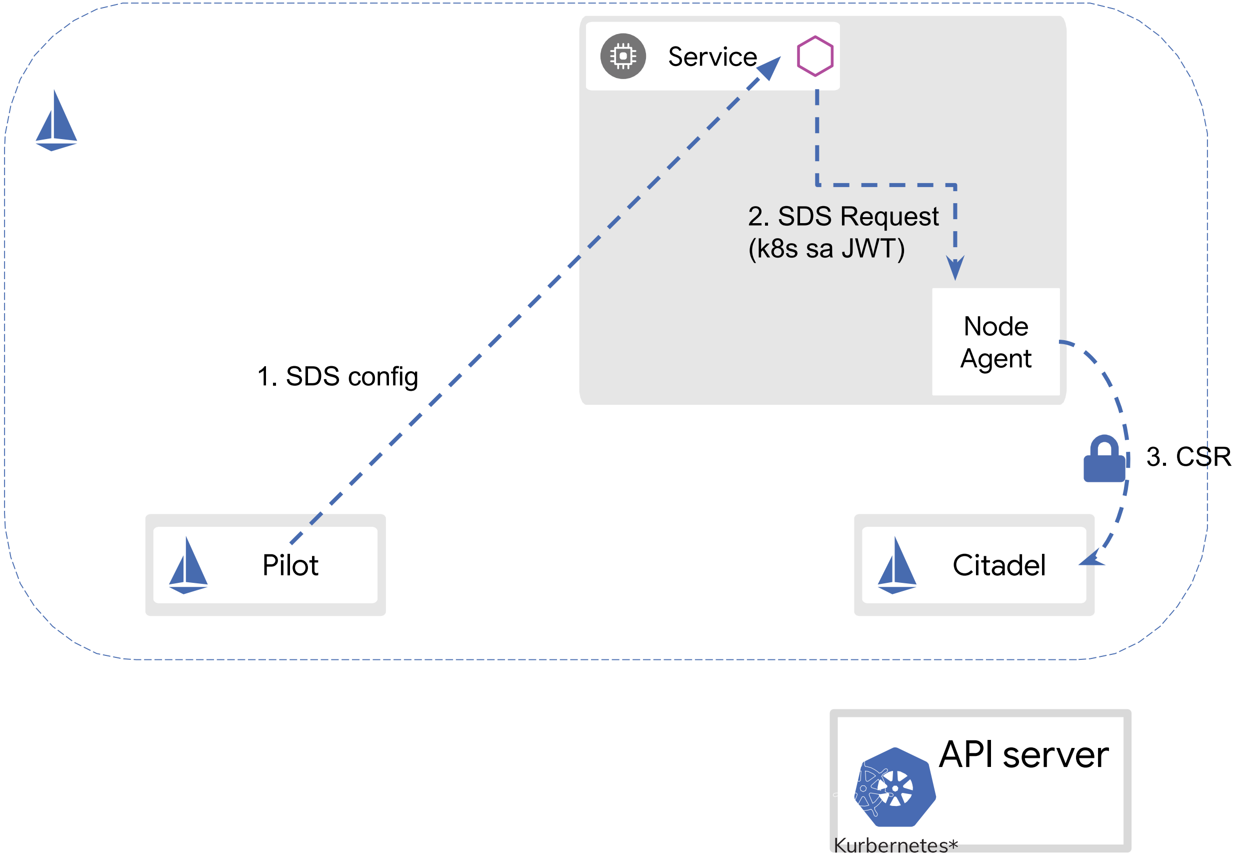


```
"cluster": {
   "name": "outbound|8000||httpbin.default.svc.cluster.local",
   "tls_context": {
    "common_tls_context": {
     "tls_certificate_sds_secret_configs": [{
          "name": "default",
          "sds_config": {
          "api_config_source": {
          "api_type": "GRPC",
          "grpc_services": [{
          "google_grpc": {
           "target_uri": "unix:/var/run/sds/uds_path",
           "call_credentials": [{
                "from_plugin": {
                 "name": "envoy.grpc_credentials.file_based_me
                  "typed_config": {
                    "secret_data": {
                        "filename": "/var/run/secrets/kubernetes.io/service
                       "header_key": "istio_sds_credentail_heade
          ...
          }],
          "combined_validation_context": {
```

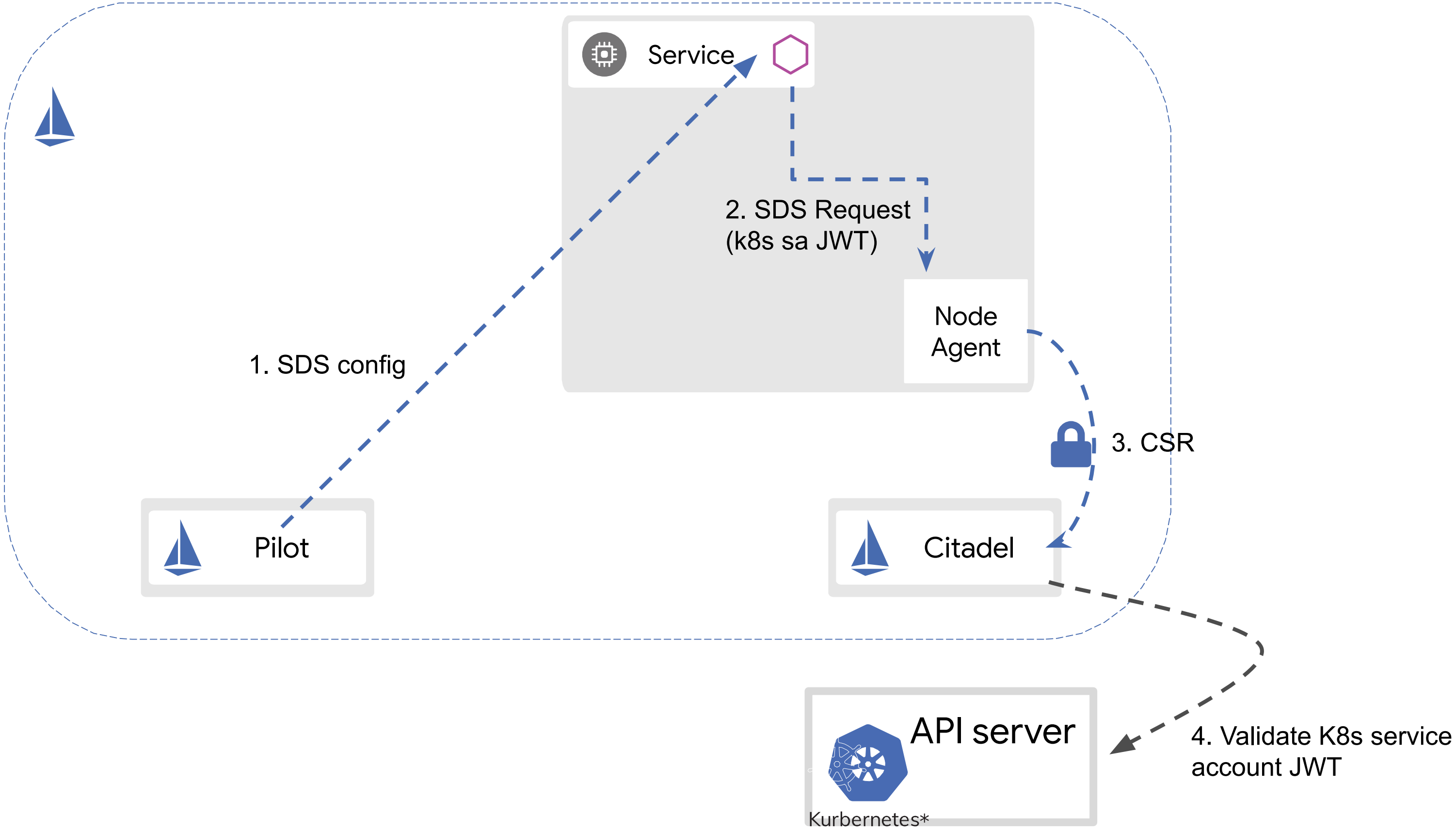
New identity provision pipeline



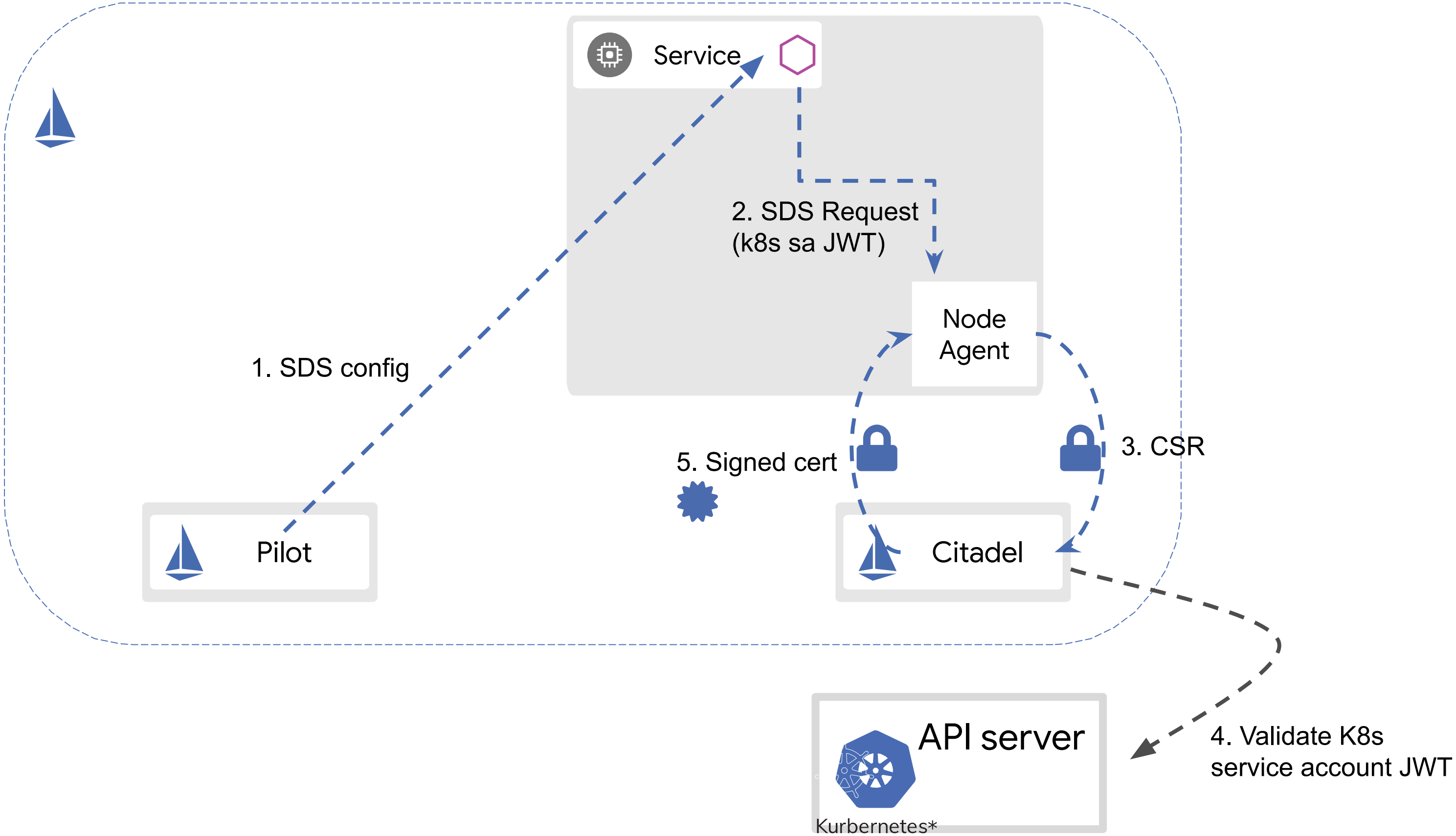
New identity provision pipeline



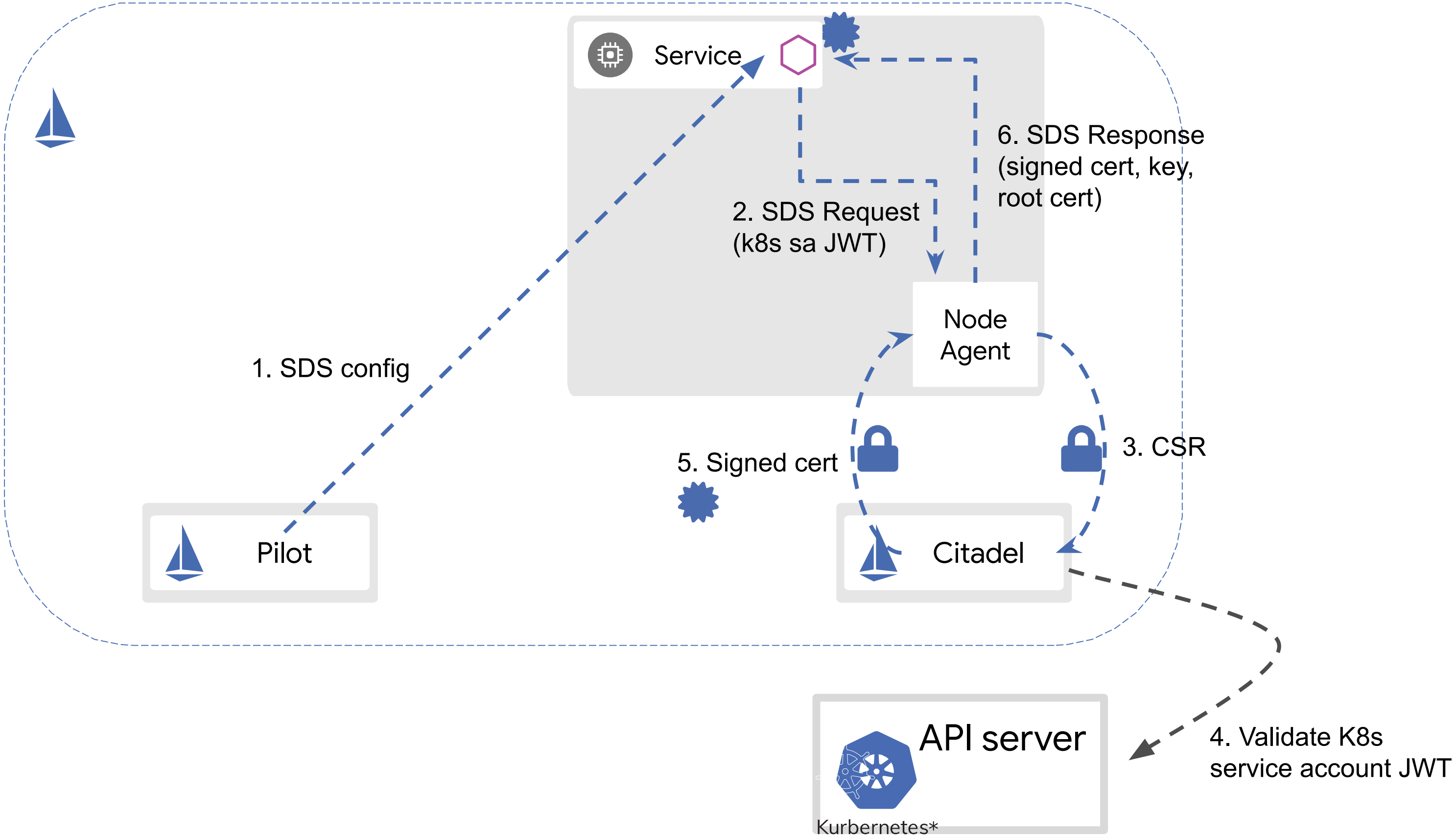
New identity provision pipeline



New identity provision pipeline

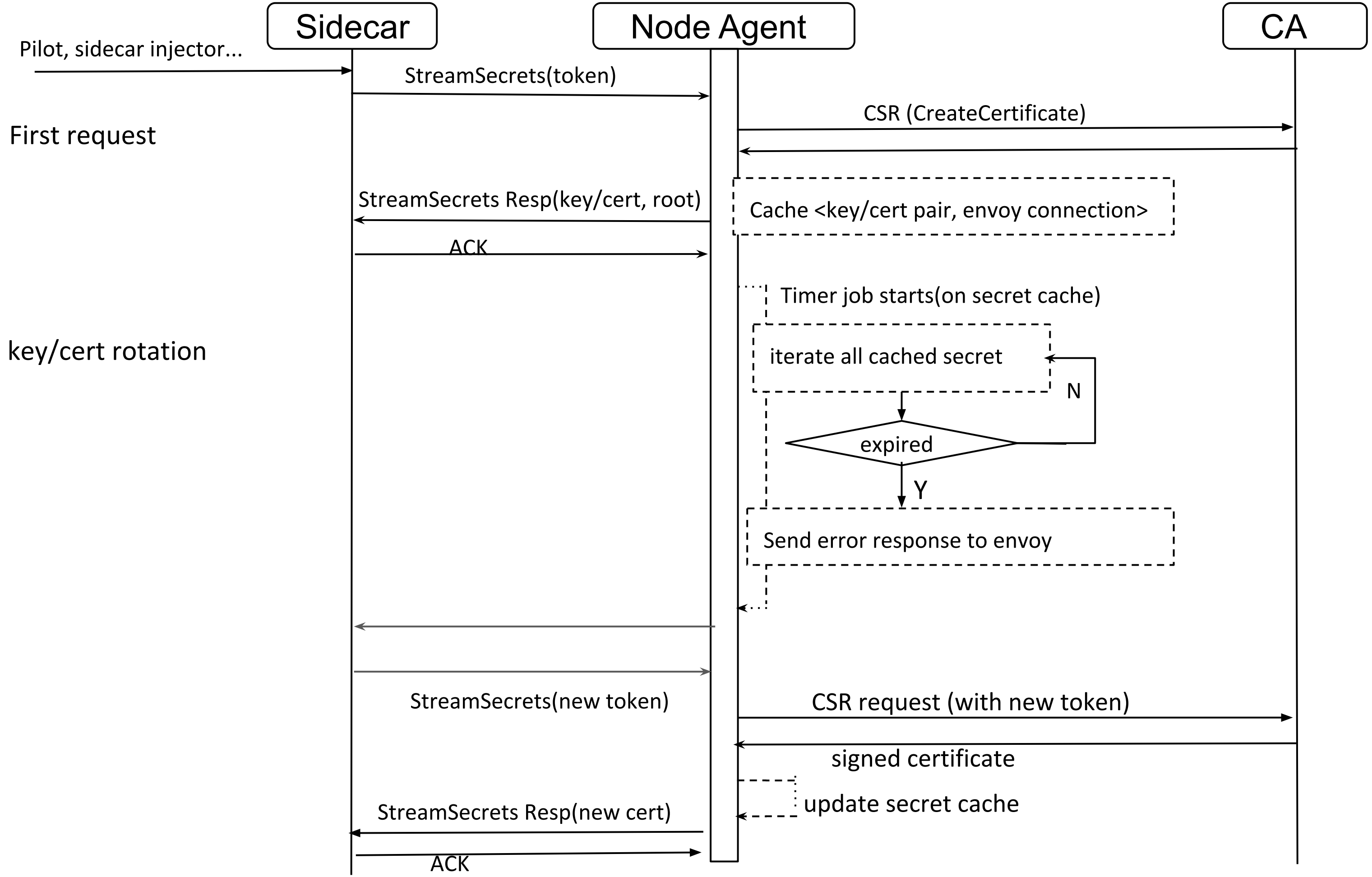


New identity provision pipeline

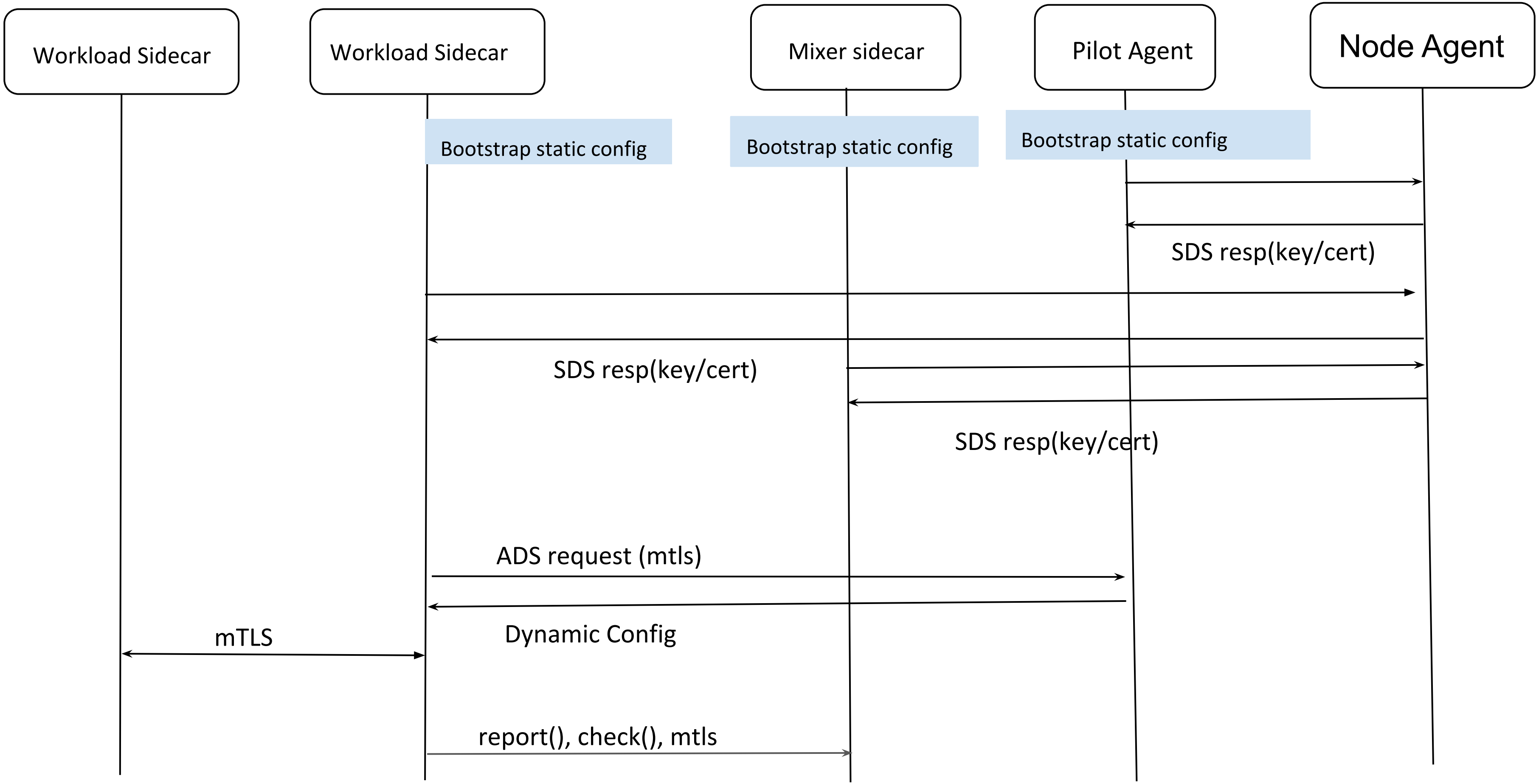


Demo

Certification rotation


Sidecar
Node Agent
CA
Pilot, sidecar injector...
StreamSecrets(token)
CSR (CreateCertificate)
First request
StreamSecrets Resp(key/cert, root)
Cache <key/cert pair, envoy connection>
ACK
Timer job starts(on secret cache)
key/cert rotation
iterate all cached secret
N
expired
Y
Send error response to envoy
StreamSecrets(new token)
CSR request (with new token)
signed certificate
update secret cache
StreamSecrets Resp(new cert)
ACK

Secure Istio Control Plane through SDS



Plug your own CA

Citadel

<config>

```
nodeagent:
  enabled: true
  image: node-agent-k8s
  env:
    CA_PROVIDER: "Citadel"
    CA_ADDR: "istio-citadel:8060"
```

<Citadel CA client>

Vault

<config>

```
nodeagent:
  enabled: true
  image: node-agent-k8s
  env:
    CA_ADDR:
"https://34.83.129.211:8200"
    CA_PROVIDER: "VaultCA"
    VAULT_ADDR:
"https://34.83.129.211:8200"
    VAULT_AUTH_PATH:
"auth/kubernetes/login"
```

<Vault CA client>

Customer CA

<define your config>

```
nodeagent:
  enabled: true
  image: node-agent-k8s
  env:
    CA_ADDR: "your CA Address"
    CA_PROVIDER: "your CA Name"
    ...//other params
```

<write your CA client>

Q&A

sample envoy config (backup slides)

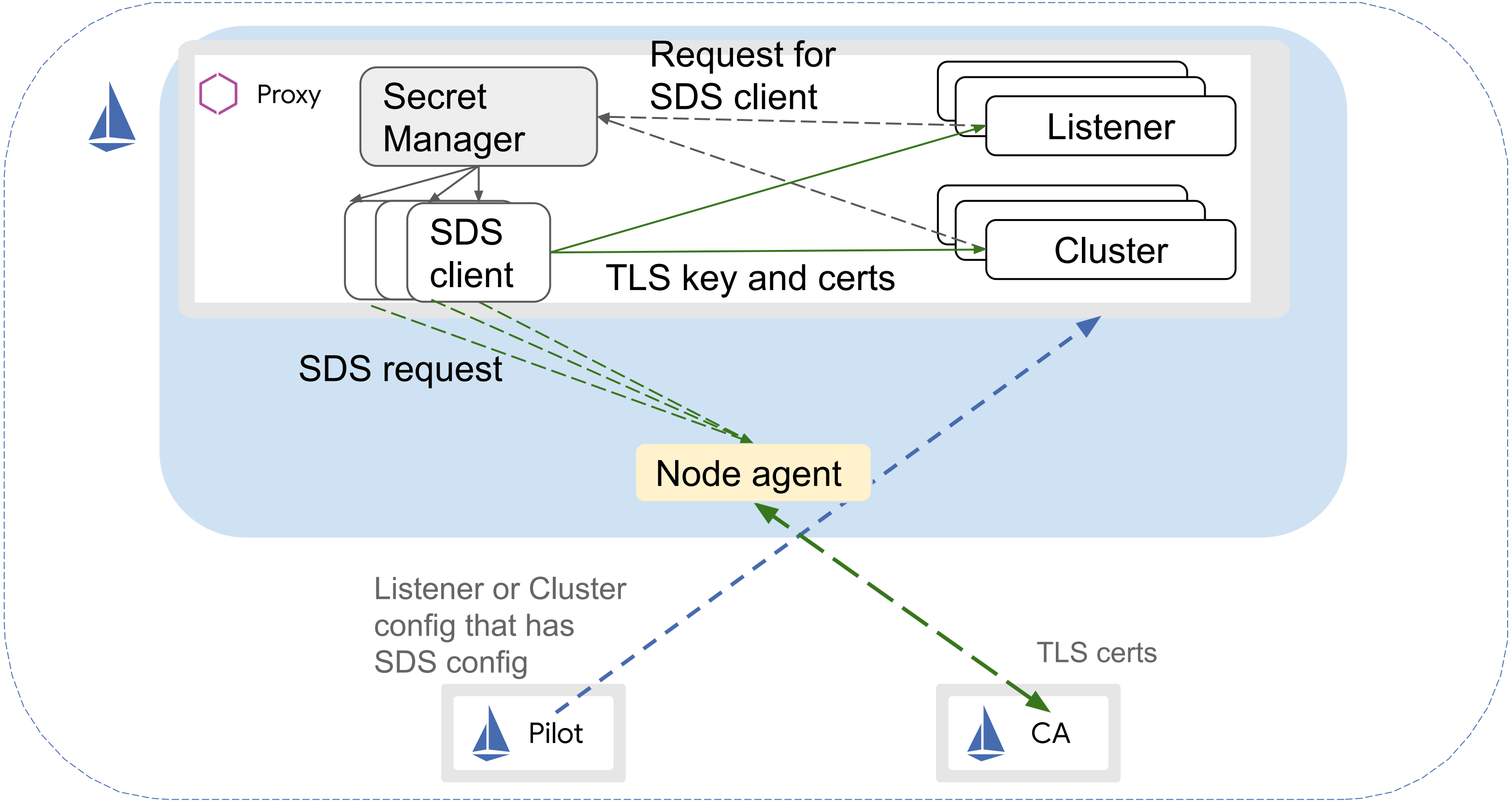
Old config(using secret file mount)

```
     "cluster": {
             "name":
"outboundǀ8000ǀǀhttpbin.default.svc.cluster.local",
             "tls_context": {
              "common_tls_context": {
                "tls_certificates": [{
                  "certificate_chain": {
                   "filename": "/etc/certs/cert-chain.pem"
                  },
                  "private_key": {
                   "filename": "/etc/certs/key.pem"

                    }
               }],
               "validation_context": {
                 "trusted_ca": {
                  "filename": "/etc/certs/root-cert.pem"
                 },
                "verify_subject_alt_name":
["spiffe://cluster.local/ns/default/sa/default"]
                ...
```

New config(using SDS)

```
"cluster": {
  "name": "outbound|8000||httpbin.default.svc.cluster.local",
  "tls_context": {
   "common_tls_context": {
    "tls_certificate_sds_secret_configs": [{
         "name": "default",
          "sds_config": {
          "api_config_source": {
          "api_type": "GRPC",
          "grpc_services": [{
          "google_grpc": {
           "target_uri": "unix:/var/run/sds/uds_path",
           "call_credentials": [{
               "from_plugin": {
                 "name": "envoy.grpc_credentials.file_based_metadata",
                  "typed_config": {
                     "secret_data": {
                         "filename": "/var/run/secrets/kubernetes.io/serviceaccount"},
                        "header_key": "istio_sds_credentail_header-bin"
          ...
          }],
          "combined_validation_context": {
```

Envoy SDS (backup slides)


Proxy
Secret Manager
Request for SDS client
Listener
Cluster
SDS client
TLS key and certs
SDS request
Node agent
Listener or Cluster config that has SDS config
TLS certs
Pilot
CA